THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 56, No. 5, May–June 1977
Printed in U.S.A.

On Speech Encryption Using Waveform Scrambling

By S. C. KAK and N. S. JAYANT

(Manuscript received September 27, 1976)

This paper presents a study of speech scramblers based on permutations of samples within an N-block. It has been found that a new family of "uniform" (U) permutations (defined by the address mapping $i \rightarrow ki$ modulo N; k prime to N) is as effective as pseudorandom (PR) permutations in destroying speech intelligibility. Analytical results show the relation between input and scrambled-signal spectra, while computer simulations compare the effects of scrambling on PAM samples and on codes based on ADM, APCM, and ADPCM. Scrambling is increasingly effective in that order, and encoding delays in ADPCM can be as low as 1 to 2 ms. Finally, scrambling has been compared with frequency inversion, which corresponds to sign-inversion in every other Nyquist-rate waveform sample.

I. INTRODUCTION AND SUMMARY OF RESULTS

Waveform scrambling permits a conceptually simple method of speech encryption. In view of a certain ambiguity of definition in the subject, we define scrambling as a reversible temporal rearrangement of waveform samples within a waveform block. Though specific scrambling methods have been well known and documented,[1,2] no general study seems to have been made to relate a temporal permutation to the associated modification of the short-term frequency spectrum. Furthermore, for a scrambling block length N, the total number of permutations is $N!$; while certain pseudorandom (PR) permutations constitute an effective subclass of these $N!$ permutations, other interesting subclasses exist, and none of these has been discussed in the literature to our knowledge. Finally, while it is known that effective speech encryption can result from scrambling several types of speech codes (from waveform coders or from linear predictive vocoders[2]), alternative speech codes have not been compared from the point of view of encryption potential. Such a comparison would be particularly useful if the comparisons included alter-

native methods for encoding speech at comparable bit rates—for example, speech waveform coding[3] using adaptive pulse code modulation (APCM), adaptive DPCM (ADPCM), and adaptive delta modulation (ADM).

This paper discusses (*i*) the effect of a permutation on the input spectrum shape; (*ii*) the generation of a new subclass of permutations characterized by a simple algorithm and by desirable spectrum modifications; (*iii*) comparison of the new permutation subfamily with shift-register-generated PR permutations[2] and with the classical technique of frequency inversion;[1] and (*iv*) comparison of scrambler performance on ADM, APCM, and ADPCM bits and on pulse-amplitude modulated (PAM) speech samples for block lengths varying from 4 to 64.

For ADM bits the sampling rate was 24 kHz. The APCM and ADPCM coders used 8-kHz samples and 3 bits per sample. Thus, the bit rate of all three coders was 24 kb/s. The PAM samples were sampled at 24 kHz for scrambling and at 8 kHz for frequency inversion; in the time domain, frequency inversion corresponds to reversing the polarity of every other Nyquist-rate PAM sample. The Nyquist frequency for the 200- to 3200-Hz speech sample used in our perceptual experiment was 8 kHz. With block-scrambling of 24-kHz samples or bits, a block length of 64 samples or bits implies a time duration of 64/24 = 2.7 ms, while a block length of four samples represents a duration of 0.17 ms.

Our study is addressed to issues of casual privacy as well as formal encryption. For privacy, the objective is to render speech as unintelligible as possible by means of one out of many possible transformations, so that only a receiver with knowledge of an inverse transformation can understand the message. There is very little change of the transformation function with time. Recorded messages in a privacy system are amenable to cryptanalysis after sufficient processing. This kind of privacy also obtains in transformations on text[1] (where, however, the secrecy requirement is potentially greater since there are no restrictions such as those in a real-time communication link).

In formal encryption, secrecy is maintained by a repeated change of the transformation procedure (referred to as the key), and the number of keys available becomes a measure of the effectiveness of the system. As a general mathematical study of encryption is already available,[4] we shall devote more attention to the specific issue of speech privacy—specifically, the problem of destroying speech intelligibility using transformations such as permutations. A formal speech encrypter would employ the strategy of switching from one permutation transformation to another in a fashion known only to the intended receiver. The technique of frequency inversion is not useful for formal encryption because there is only one key associated with it. On the other hand, we can use the technique of adding a masking signal,[1] such as modulo 2 addition

of a pseudorandom binary sequence,[2] for formal speech encryption in real time. We have compared scrambling with this technique and have found that for speech applications where some processing delay is acceptable, temporal scrambling may be preferable. This is because the key information in scrambling is very small, and it takes very little information to transmit changes of keys.

The rest of this section summarizes the results of this paper. The novel subclass of permutations proposed in this paper is defined by the simple mapping of sample-position r into position s, with

$$s = k_1 \times r \text{ (modulo } N\text{)}$$
$$r = 1, 2, \cdots N, \tag{1a}$$

where k_1 is prime to N and N is the number of samples in the waveform block that is being scrambled.

We refer to the one-to-one mapping above as uniform, or U, permutations (see Section II). The set of U permutations increases faster with block length N than the corresponding set of PR permutations; for $N = 8$, it is larger by a factor of about 2, while for $N = 128$, this factor is nearly 7.

The analytical results presented in the paper include an algorithmic relation between a permutation and the modification it produces in the DFT spectrum. Thus, if permutations of time samples are characterized by a matrix P (see Section II) and if F is the standard DFT matrix [given by (5)], the transformation T of DFT samples is expressed by

$$T = FPF^{-1}. \tag{2}$$

For example, it has been shown that U permutations produce a uniform scrambling in the DFT spectrum as well. Descrambling is performed by another U permutation:

$$r = k_2 s \text{ (modulo } N\text{)}$$
$$\text{if } k_1 k_2 = 1 \text{ (modulo } N\text{)}. \tag{1b}$$

Thus, for $N = 32$, if scrambling is done with $k_1 = 7$, descrambling will need $k_2 = 23$.

Perceptual comparisons have been made of U permutations, PR permutations, and frequency inversions for PAM, ADM, APCM, and ADPCM samples. Results indicate the following increases in efficiency:

(*i*) PAM < ADM < APCM < ADPCM for scrambling with a given block length N.

(*ii*) ADM < PAM < APCM < ADPCM for frequency inversions.

The nature of the encrypted waveform is different between the first two and the last two codes, due conceivably to the fact that in PCM and DPCM, different bits have different weights associated with them. For

scrambling, U permutations performed at least as well as PR permutations, and, for one case of $N = 32$, in fact did better in informal perceptual comparisons. For longer block lengths, no meaningful comparisons could be made between the two scramblers, because intelligibility was totally destroyed in each case (except with PAM samples). We have also found that if the scrambling delay is sought to be less than about 2 ms, it becomes essential to use APCM or ADPCM codes.

This paper is arranged as follows: Section II presents analytical results on permutation transformations and compares U and PR permutations. Section III briefly discusses the classical technique of frequency inversion. Section IV summarizes perceptual results and spectrograms from a computer simulation. Section V provides a heuristic comparison of masking and scrambling as alternative techniques for encryption.

II. PERMUTATION TRANSFORMATIONS

2.1 Permutation matrices

The total number of permutations possible on a block of N samples is $N!$. If we also included the possibility of sign changes of the samples, this total is increased to $2^N N!$. As sign changes are easy to implement, a general study of such transpositions should be useful. In this paper, however, we restrict ourselves to simple permutations only, except for the special case of sign changes of alternate samples (Section III). This corresponds to classical frequency inversion, and is therefore of interest for comparison with permutations (scrambling).

Temporal sample permutations can be characterized by a permutation operator P that is a matrix of ones and zeros. Thus, for a block of $N = 5$ samples, if bit positions 0, 1, 2, 3, and 4 are permuted to 0, 1, 4, 2, and 3, we can denote the permutation by

$$[0, 1, 2 \rightarrow 4 \rightarrow 3 \rightarrow 2]$$

or by the permutation matrix

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

Expressing P as $[2 \overset{P}{\rightarrow} 4 \overset{P^2}{\rightarrow} 3 \overset{P^3}{\rightarrow} 2]$ gives the order of the cyclic group defined by P directly; since P^3 brings 2 back to 2, the order is 3. In other words, P^3 is an identity matrix I.

If a permutation is defined in terms of more than one cycle, it can be seen that it has an order equal to the least common multiple (lcm) of (order of cycle 1, order of cycle 2, $\cdots$). For example, the permutation [0

$\rightarrow 3 \rightarrow 5 \rightarrow 0, 2 \rightarrow 4 \rightarrow 6 \rightarrow 1 \rightarrow 2]$ has an order = lcm (3, 4) = 12, and this permutation is characterized as a $P(3, 4)$ permutation. The number of repeated $P(3, 4)$ permutation operations that will bring us back to an original unpermuted sequence is 12. (The permutation in the earlier example was of the $P(1, 1, 3)$ type; in that case, three successive $P(1, 1, 3)$ operations would lead to the unpermuted original sequence.) In general, if

$$P: P(p_1, p_2, p_3, \cdots),$$

where

$$p_1 + p_2 + p_3 + \cdots = N, \tag{3a}$$

the number of repeated P permutations that would result in the original unpermuted sequence [equivalently, the number of distinct mappings $L(P)$ that can be generated using P] is

$$L(P) = \text{lcm}\ (p_1, p_2, p_3 \cdots). \tag{3b}$$

2.2 Effect of permutations on frequency spectrum

Let $x(n)$, $(n = 0, 1, \cdots, N-1)$ be the discrete input block of length N, which is to be permuted. Let $X(m)$, $(m = 0, 1, \cdots, N-1)$ be its DFT, and let x and X denote corresponding column vectors $[x(0), x(1), \cdots x(N-1)]'$ and $[X(0), X(1), \cdots X(N-1)]'$, respectively. Note that

$$X = Fx, \tag{4}$$

where

$$F = \begin{bmatrix} 1 & 1 & 1 & 1 & . & . & 1 \\ 1 & W & W^2 & W^3 & . & . & W^{N-1} \\ 1 & W^2 & W^4 & W^6 & . & . & W^{2(N-1)} \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 1 & W^{N-1} & W^{2(N-1)} & . & . & . & W^{(N-1)^2} \end{bmatrix} \tag{5}$$

with the standard DFT identities

$$W = \exp\left(-j\frac{2\pi}{N}\right); \quad W^N = 1$$

and (6)

$$\sum_{r=0}^{N-1} W^r = 0.$$

Let X_P, the DFT of the permuted sequence, be described as a transformation of X:

$$X_P = TX = TFx. \tag{7a}$$

Also, in the manner of (4),

$$X_P = FPx. \tag{7b}$$

From (7a) and (7b),

$$FP = TF, \text{ or}$$

$$T = FPF^{-1} \tag{8a}$$

and

$$P = F^{-1}TF, \tag{8b}$$

where the inverse of F can be seen to be

$$F^{-1} = \frac{1}{N}\begin{bmatrix} 1 & 1 & 1 & . & . & 1 \\ 1 & W^{-1} & W^{-2} & . & . & W^{-(N-1)} \\ 1 & W^{-2} & W^{-4} & . & . & W^{-2(N-1)} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 1 & W^{-(N-1)} & . & . & . & W^{-(N-1)^2} \end{bmatrix}. \tag{9}$$

Theorem: *If P is represented in terms of a, b, c, d in the earlier characterization* $[0 \to a, 1 \to b, 2 \to c, 3 \to d, \cdots]$,

$$T_{rs} = \frac{1}{N}[W^{-as} + W^{r-bs} + W^{2r-cs} + W^{3r-ds} + \cdots], \tag{10}$$

where, as usual, further simplifications can be made using the relations in (6).

For a proof of this theorem, see the Appendix.

Example 1: Let P_1: $[0 \to 2 \to 1 \to 3 \to 0]$. That is, $N = 4$, $W = -j$, $W^2 = -1$, $W^3 = j$, and $W^4 = 1$; and $a = 2$, $b = 3$, $c = 1$, and $d = 0$.

$$T_1 = \begin{bmatrix} -1/2 & (1+j)/2 & j/2 & 0 \\ (j-1)/2 & 0 & (-j-1)/2 & 0 \\ -j/2 & (1-j)/2 & -1/2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

$$T_1^2 = \begin{bmatrix} 0 & 0 & -j & 0 \\ 0 & -1 & 0 & 0 \\ j & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \quad T_1^4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

It is seen that T_1 produces a transformation where DFT values are added together with different weights and phase shifts; while T_1^2 simply changes the positions of varying DFT values with varying phase shifts without modifying magnitudes. The fact that $T_1^4 = I$ could also be deduced by noting that in the time domain, $P^4 = I$ (because the order of P is 4).

Example 2: Let

$$P_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & . & . & . & 0 & 0 \\ 0 & 0 & 0 & 0 & . & . & . & 0 & 1 \\ 0 & 0 & 0 & 0 & . & . & . & 1 & 0 \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ 0 & 0 & 1 & 0 & . & . & . & 0 & 0 \\ 0 & 1 & 0 & 0 & . & . & . & 0 & 0 \end{bmatrix}. \tag{11a}$$

P_2 corresponds to $[0 \rightarrow 0, 1 \rightarrow N-1, 2 \rightarrow N-2 \cdots]$. Hence, from (10),

$$N \cdot T_{rs} = 1 + W^{r-s(N-1)} + W^{2r-s(N-2)} + \cdots$$

$$= 1 + W^{r+s} + W^{2(r+s)} + \cdots = (1 - W^{(r+s)N})/(1 - W^{r+s})$$

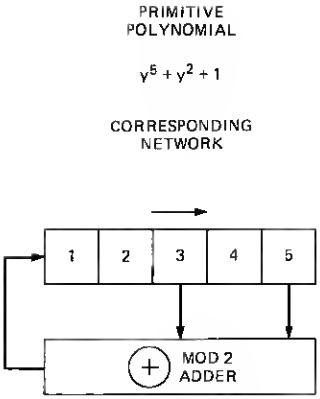
using (6); since the numerator in the above sum is always zero, T_{rs} is nonzero if and only if the denominator is zero, or if $s = MN - r; M = 0, 1, 2 \cdots$; in this case, $NT_{rs} = N$, or $T_{rs} = 1$. Consequently,

$$T_2 = P_2. \tag{11b}$$

In words, local (intra-block) time-inversion (excluding the first sample) causes a corresponding DFT inversion (excluding the first sample). Notice that an unchanged first sample in the DFT vector represents an unchanged zero-frequency value of the input spectrum. (We emphasize here that DFT inversion does not represent analog frequency inversion, which will be discussed in Section III. The DFT spectrum is symmetrical about its middle point; inverting it is a perceptually trivial operation (see Section IV), and DFT inversion does not represent a useful means for speech encryption.

2.3 *Pseudorandom permutations*

A pseudorandom (PR) scrambling of samples within a block of length N is achieved by assigning to each sample a new address A ($A = 1$, or 2, or $3, \cdots$, or N) determined by the state of a maximal-length shift-register arrangement. The theory and design of maximal-length sequences is well documented.[5,6,7] We shall therefore only provide a constructive recapitulation in this paper. The approach is to start with a shift register whose length is $D = \log_2 N$ (assume that the block length N is a power of 2 and



NO.	STATE OF SHIFT REGISTER	DECIMAL EQUIVALENT
1	00001	1
2	10000	16
3	01000	8
4	00100	4
5	10010	18
6	01001	9
7	10100	20
↑	-----	↑
INPUT ADDRESSES	REPETITION PERIOD = 31	OUTPUT ADDRESSES

Fig. 1—Pseudorandom scrambler with a five-stage shift register.

that elements in the register are either 1 or 0). The next step is to select a primitive polynomial $Q(y)$ of degree D and to include stage $D - d$ in the register ($d = 0, 1, \cdots D - 1$) in an EXCLUSIVE OR feedback (modulo 2-add) arrangement if the coefficient of y^d in $Q(y)$ is nonzero. The resulting network now generates a succession of $2^D - 1 = N - 1$ nonzero states in the shift register at successive clock times, after which the cycle repeats, starting once again with the original initial state of the shift register. The number of nonzero states in the cycle is identically equal to the repetition period $N - 1$ of the cycle. Consequently, the $N - 1$ states of the shift-register (specifically, their decimal equivalents) can be used as PR addresses for a block of $N - 1$ input samples in a one-to-one mapping of addresses. If the input block has N rather than $N - 1$ samples (because of the frequent requirement that N be a power of 2), the address of the Nth sample is usually left unaltered by the scrambler. Such simplification is, however, not mandatory, and appropriate manipulations that scramble all N samples are quite conceivable.

Figure 1 illustrates the scrambler design for the example of $D = 5$ and $N = 31$, as defined by a primitive polynomial $Q_5(y) = y^5 + y^2 + 1$. We see how input samples $(1, 2, 3, 4, 5, 6, 7, \cdots)$ get scrambled into PR positions $(1, 16, 8, 4, 18, 9, 20, \cdots)$. We can verify easily that the mapping is one to one for the 31 sample values in an input block. The PR scrambling of this example is illustrated more completely in the permutation matrix in Section 2.5 (Fig. 3a).

It is clear that in the arrangement of Fig. 1, the use of a different initializing sequence (other than 00001) can lead to a totally different mapping of sample addresses. There would be $N - 1$ nonzero initializations corresponding to every given $Q_5(y)$. In addition, the number of primitive polynomials $Q_5(y)$ of degree 5 is 6, and this implies a further increase in the total number of possible mappings.

Table I — List of primitive polynomials

Degree D	Typical Primitive Polynomial $Q_D(y)$	Number of Primitive Polynomials $L(D)$ of Degree D
1	$y + 1$	1
2	$y^2 + y + 1$	1
3	$y^3 + y + 1$	2
4	$y^4 + y + 1$	2
5	$y^5 + y^2 + 1$	6
6	$y^6 + y + 1$	6
7	$y^7 + y + 1$	18
8	$y^8 + y^4 + y^3 + y^2 + 1$	16
9	$y^9 + y^4 + 1$	48
10	$y^{10} + y^3 + 1$	60
11	$y^{11} + y^2 + 1$	176
12	$y^{12} + y^6 + y^4 + y + 1$	144

Table I lists, for $D = 1$ to 12, a typical set of primitive polynomials, and also the number of primitive polynomials $L(D)$ for each D. Note, for example, that a 12-stage shift register with an EXCLUSIVE OR feedback network involving stages 12, 11, 8, and 6 ($D - d$; $d = 0, 1, 4$, and 6) provides one of 144 possible bases for a scrambler that would operate on an input block of $2^{12} = 4096$ samples.

The possibility of distinct scrambler mappings (number of keys) as defined by different initializations and/or different primitive polynomials is an important consideration from the point of view of the average descrambling time needed for an eavesdropping code-breaker. Let us note formally, then, that the number of keys K for a PR permutation of N sample blocks ($N = 2^D$) is

$$K_{PR} = (N - 1)L(\log_2 N). \tag{12}$$

The first term of the product in (12) represents the total number of shift-register initializations, and the second term gives the number of primitive polynomials (Table I) of degree $D = \log_2 N$. Numerical values of K_{PR} are discussed in Section 2.5.

2.4 Uniform permutations

We now discuss the new subclass of scramblers defined by an output/input (s-r) mapping indicated in (1a):

$$s = k_1 r \text{ (modulo } N)$$

$$r = 1, 2, \cdots N; \quad k_1 \text{ is prime to } N. \tag{13a}$$

It can be verified that k_1 should be prime to the block-length N for the

UNIFORM ONE–TO–ONE MAPPING

$r \longrightarrow 7r$ (MOD 32)

$r = 1, 2, \cdots 32$

INPUT ADDRESS	OUTPUT ADDRESS
1	7
2	14
3	21
4	28
5	3
6	10
7	17
—	—
r	7r

Fig. 2—Uniform scrambler ($r \rightarrow 7r$ mod 32).

mapping (13a) to be one to one. Figure 2 illustrates these permutations for the example of $N = 32$ and $k_1 = 7$. The complete permutation matrix is discussed in Section 2.5 (Fig. 3b), where it may be noted that the 1s in the matrix are spread uniformly in the 32 by 32 matrix—hence, the name uniform permutations, or U permutations.

The descrambling of a U permutation is achieved by means of another (inverse) U permutation so that $k_1k_2 = 1$ (modulo N):

$$r = k_2 s \text{ (modulo } N\text{)}$$
$$s = 1, 2, \cdots N \tag{13b}$$

and $k_1k_2 = 1$(modulo N). The condition that $k_1k_2 = 1$ follows from the requirement that $k_2s = k_2(k_1r) = r$, for all r. Thus, for $N = 32$, if scrambling is done with $k_1 = 7$, descrambling will require $k_2 = 23$ in order that $k_1k_2 = 161 \bmod 32 = 1$.

2.4.1 Choice of k_1

The value of k_1 in (13a) is an interesting issue. We do not offer any rigorous criterion for optimizing k_1, and we believe in general that there is no single optimum for practical applications where, in fact, we use different k_1 values as different keys in encryption. However, there are three observations regarding k_1 that are worth noting.

First, a reasonable scrambling procedure is one that distributes N 1s uniformly in an N by N permutation. For simplicity, let N be a perfect square. The uniform distribution problem is then one of dividing the N by N matrix into a number N of $\sqrt{N}$ by $\sqrt{N}$ submatrices, and to place one and exactly one of the N 1s in the center of each submatrix. Adjacent 1s would then be separated by a distance equal to the side of the square submatrix. Indeed, this distance $\sqrt{N}$ would correspond to the uniform mapping parameter k_1. On the other hand, N in general need not be a

perfect square, and $\sqrt{N}$ need not be an integer. Furthermore, k_1 should be prime to N for one-to-one mapping, and if $\sqrt{N}$ should indeed be an integer, it will not be prime to N. All that can be said in conclusion, then, is that in the absence of a better criterion, a value of k_1 that is close to $\sqrt{N}$, and at the same time is prime to N, would represent a reasonable design value.

A second viewpoint on the value of k_1 relates to adjacent sample correlations. The U permutation causes input samples separated by a distance k_1 to be brought together. Thus, if input samples separated by the distance k_1 are uncorrelated, then the U permutation will convert the original N sequence to one in which adjacent samples tend to be uncorrelated. In a subsequent section, we discuss cross-correlations between input and scrambled sequences (and corresponding cross-spectra).

The third observation is that there is an interesting geometrical interpretation of k_1. It is related to the slopes of the zig-zag straight line loci obtained by joining successive 1s in the permutation matrix while scanning it from top to bottom (increasing r).

Finally, notice that the special value of $k_1 = N - 1$ corresponds to DFT inversion (local time inversion, as discussed earlier).

2.4.2 Number of keys in U permutations

It can be seen that the number of keys (distinct mappings K for U permutation is a product of the form

$$K_{\mathrm{U}} = NG(N), \tag{14}$$

where $G(N)$ is used to denote the number of k_1 values that are prime to the block length N, and N is the number of cyclic shifts (translations by one sample) of an input block prior to permutation via a given permutation matrix (which is equivalently expressed as the number of ways of selecting the first row in vertical cyclic shifts of the permutation matrix). It can be verified that the following values of G apply for the special cases where N is prime or a power of two:

$$G(N) = N - 2 \text{ if } N \text{ is prime}$$
$$G(N) = N - D - 1 \text{ if } N = 2^{\mathrm{D}}. \tag{15}$$

Numerical values of K_{U} are discussed in Section 2.5.

2.4.3 Effect of U permutations on frequency spectrum

The effect of U permutations on input spectrum follows immediately from an earlier relation (10). For U permutations, then, the DFT transposition matrix T is characterized by

$$NT_{rs} = 1 + W^{r-sk_1} + W^{2r-2sk_1} + \cdots. \tag{16a}$$

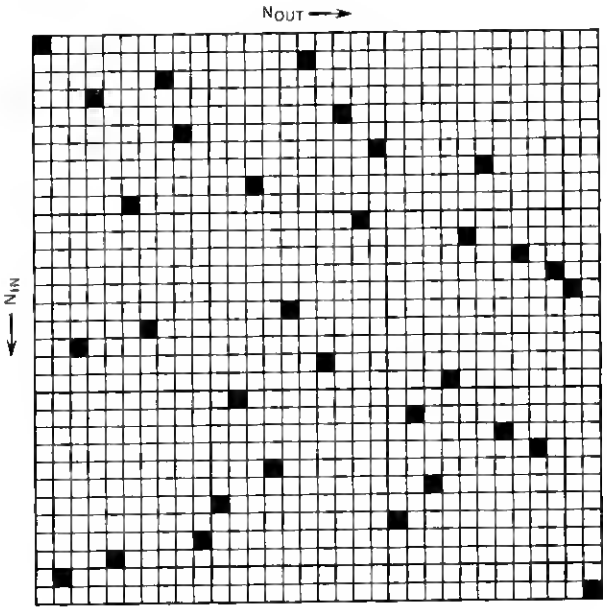


Fig. 3(a)— Pseudorandom permutation matrix.

Summing the geometric series above,

$$T_{rs} = \frac{1}{N}\left[\frac{1 - W^{rN - sk_1N}}{1 - W^{r - sk_1}}\right].$$

Because $W^N = 1$ [from (6)], the numerator above is always zero. A nonzero T_{rs} therefore requires that the denominator is zero as well:

$$W^{r-sk_1} = 1, \quad \text{or}$$

$$r - sk_1 = 0, \quad \text{or}$$

$$s = r/k_1. \tag{16b}$$

It can be verified from (16a) that this condition will make $T_{rs} = 1$. Furthermore, if (16b) should hold for all r, one requires that s/r should be independent of r. That is,

$$s = k_3 r, \tag{16c}$$

so that from (16b),

$$k_3 r = r/k_1 \text{ or}$$

$$k_1 k_3 = 1 \text{ (modulo } N). \tag{16d}$$

In other words, uniform permutation in the time-domain causes a uniform permutation in the frequency domain. Thus, if the U permu-

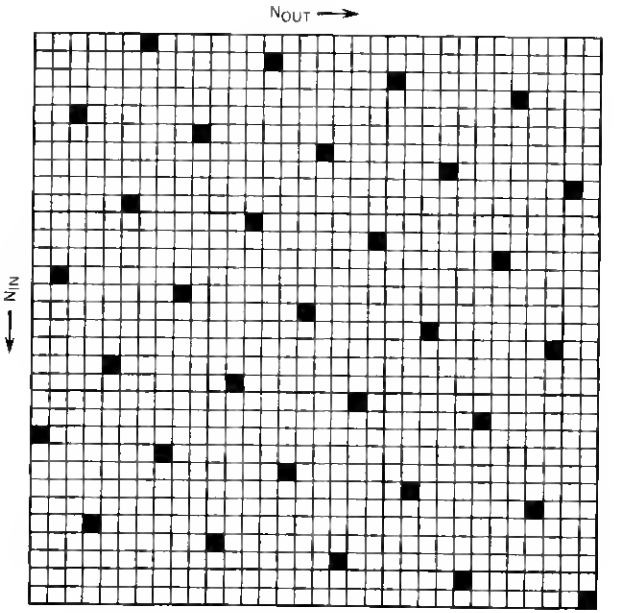


Fig. 3(b)—Uniform permutation matrix.

tations use $k_1 = 7$, $k_3 = 23$ from (16d); and the mapping (16c) (which denotes nonzero T_{rs} values) indicates the following frequency transpositions: $[0 \rightarrow 0, 1 \rightarrow 23, 2 \rightarrow 14, 3 \rightarrow 5, \cdots$ or in general $t \rightarrow 23t$ (modulo 32) for $t = 0, 1, \cdots 31]$. Notice, once again, the special example of DFT inversion and the corresponding local time inversion; these are expressed by the mappings $k_3 = N - 1$ and $k_1 = N - 1$, respectively.

2.5 Comparisons of PR and U permutations

2.5.1 Permutation matrices

Figures 3a and b are typical permutation matrices for PR and U permutations. As in Fig. 1, the PR matrix of Fig. 3a is based on a fifth-order primitive polynomial $Q_5(y) = y^5 + y^2 + 1$, a beginning shift-register state of 00001 and a mapping of position 32 into itself. Following Fig. 2, the U matrix of Fig. 3b is based on a value of $k_1 = 7$.

2.5.2 Number of keys in encryption

Table II lists illustrative values of the number of keys K_{PR} and K_U, as obtained from (12) and (14), respectively. Note that with both PR and U mappings, not all of the total number of permutations can be equally effective in destroying speech intelligibility. A good example of a per-

Table II — Number of keys in PR and U permutations

N (number of samples in block)	$K_{PR} = (N-1)L(\log_2 N)$	$K_U = NG(N)$
8	14	32
16	30	176
32	186	832
64	378	3648
128	2286	15360
256	4080	63232

ceptually uninteresting permutation is DFT inversion* (see Section IV). In any case, on the basis of Table II, N_U increases much faster with N than N_{PR}. For this reason, U permutations are potentially more attractive candidates, *a priori*, for formal encryption.†

2.5.3 Effects on frequency spectrum

Assessments of speech encryption techniques should be really based on perceptual testing; Section IV describes such testing and also provides interesting speech spectrograms. Meanwhile, it is instructive to compare PR and U permutations on the basis of their effects on illustrative input spectra.

Figure 4 provides these comparisons. Figures 4 through 6 use linear low-pass, high-pass, and mid-pass models for input spectra, while the input spectrum of Fig. 7 is a simplified three-pole model that could be an example of the short-term spectrum of a voiced speech sample. Note that both PR and U permutations are effective in distorting input spectra; in fact, the tendency in each case is to whiten the spectrum. However, the whitening in PR permutations is both global and local, while U permutations produce whitening only in a global (average) sense. In other words, PR-scrambled spectra are smoother in terms of adjacent sample transitions in the output spectrum. This is due to the fact that entries in the DFT-transposition matrix T, eqs. (7a) and (8a), have varying weights in PR permutation. With U permutations, on the other hand, the only nonzero entries in T are ones; the effect is simply one of rearranging input DFT samples without any magnitude weighting.

2.6 Cross-correlations between input and scrambled samples

Spectral distortions that provide optimal speech encryption are difficult to specify; and further, they are likely to be input-spectrum de-

* It is interesting that DFT inversion is ineffective despite the fact that the associated permutation (time inversion) is characterized by a large input-output "distance" $\sum_{r=1}^{N} |r - s(r)|$.

† Another advantage of U permutations is that N need not be a power of 2.

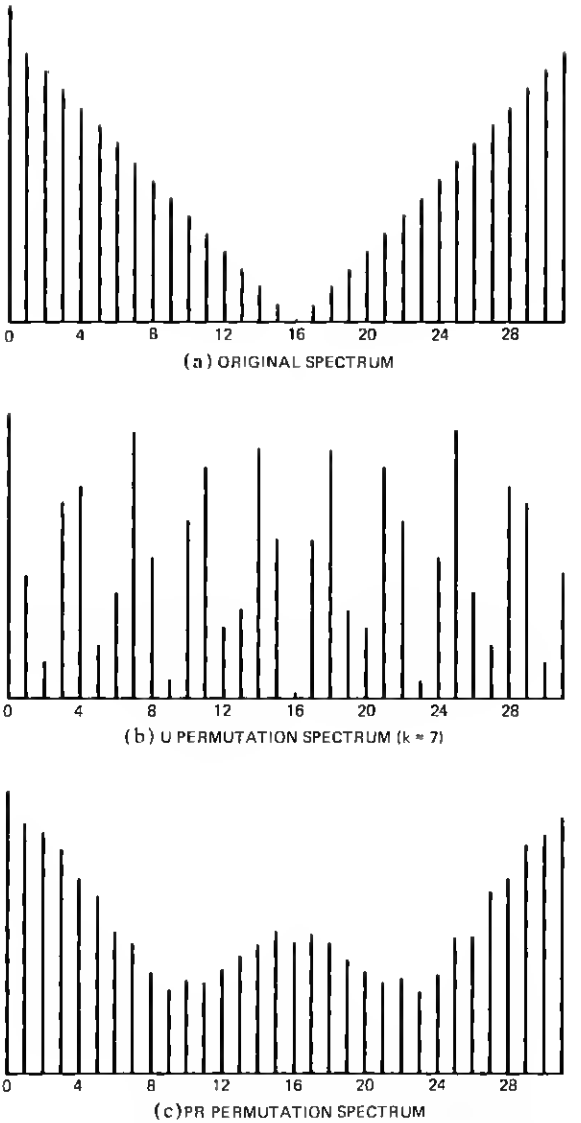


Fig. 4—Effects of PR and U permutations on low-pass spectrum.

pendent. It appears reasonable, however, that the whitening effects in Figs. 4 through 7 are very desirable for encryption. A different criterion for encryption is the decorrelation of input and scrambled spectra; spectral whitening is in general a sufficient, but not necessary, condition for such decorrelation.

Let us define a correlation measure C by $x'x$, which represents the dot

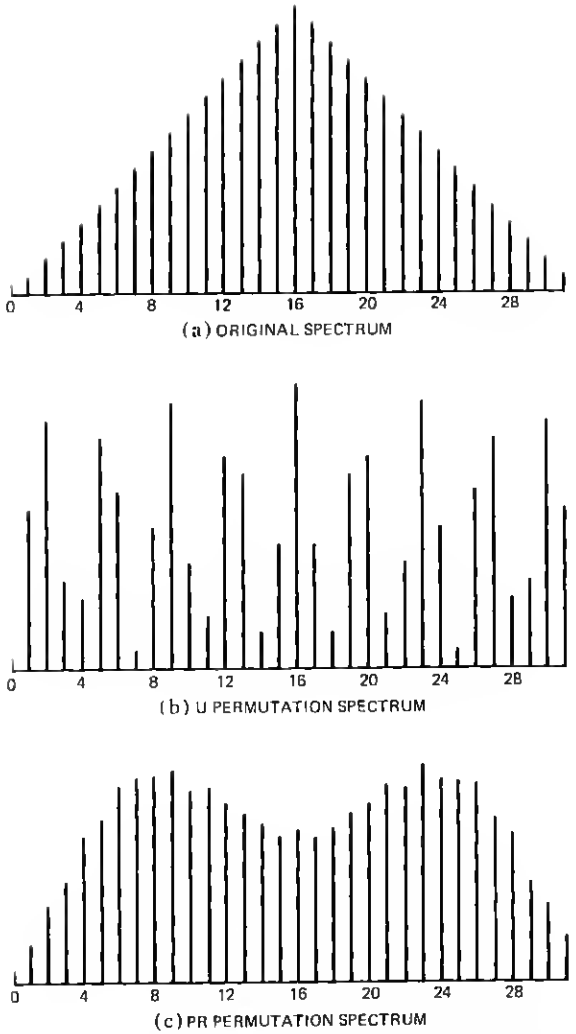


Fig. 5—Effects of PR and U permutation on high-pass spectrum.

product of an input sequence with itself. Correlation between scrambled and original sequences is $x'_P x$, where the subscript P indicates permutation or scrambling. A corresponding correlation between input and scrambled spectra is $X'_P X^*$, where X^* is the complete conjugate of the input spectrum X. Notice that

$$X'_P X^* = (FPx)'(Fx)^* = (Px)'F'F^*x.$$

It can be shown from (5) that $F'F^* = IN$, where I is an identity matrix. Consequently, spectral and time-domain cross-correlations between

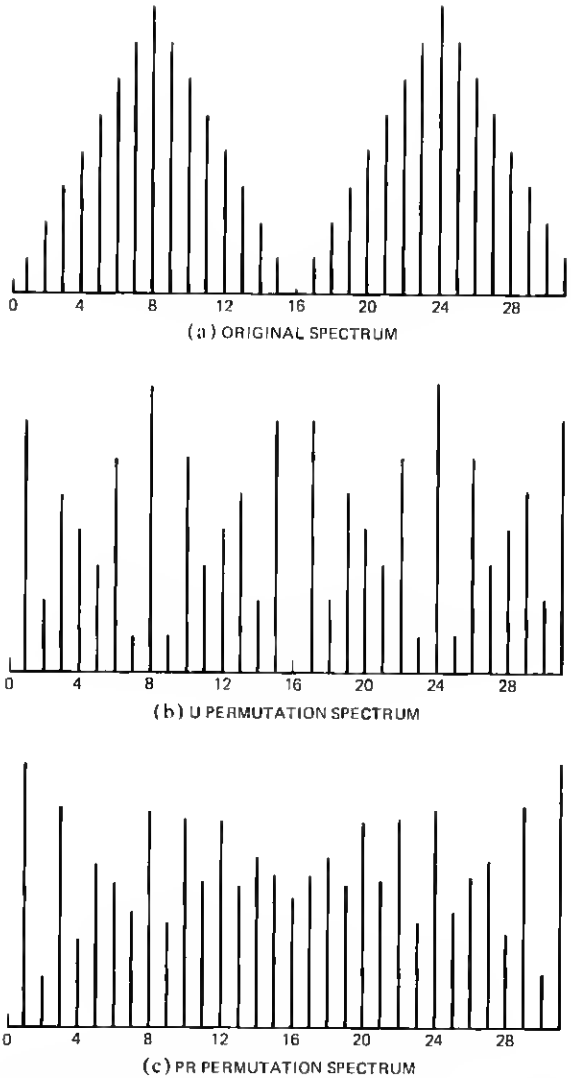


Fig. 6—Effects of PR and U permutations on mid-pass spectrum.

input and scrambled samples are related by

$$X'_P X^* = N(Px)'x. \tag{17}$$

In other words, spectral decorrelation requires that the permuted and the original sequences are themselves orthogonal or uncorrelated. The significance of negative cross-correlations is not always clear. In fact, in at least one instance, a negative cross-correlation of -1 between input and scrambled spectra is known to be perceptually suboptimal. This is

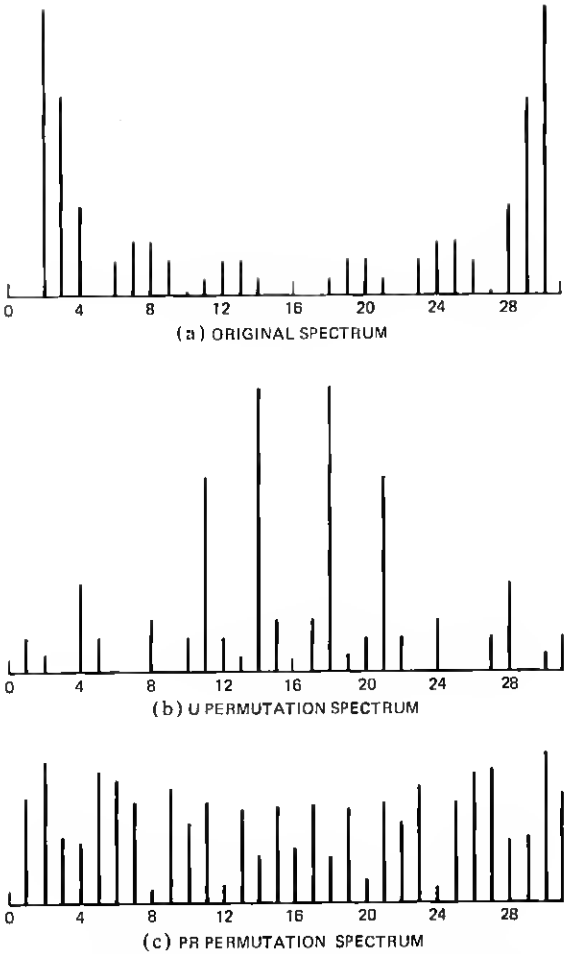


Fig. 7—Effects of PR and U permutations on an example of a voiced speech spectrum.

the example of frequency inversion (Section III); however, it has been reported that listeners can be trained to understand frequency-inverted speech[1]. It would appear then that scramblers that cause the cross-correlations in (17) to approach zero are, in general, very desirable. The case of zero correlation is realized for U permutation, for example, if input message samples that are separated by a distance of k_1 are uncorrelated.

2.7 Permutations of binary sequences

In practical speech-encryption techniques, we scramble speech-carrying bits in digital codes such as PCM, DPCM, and DM[3] rather than

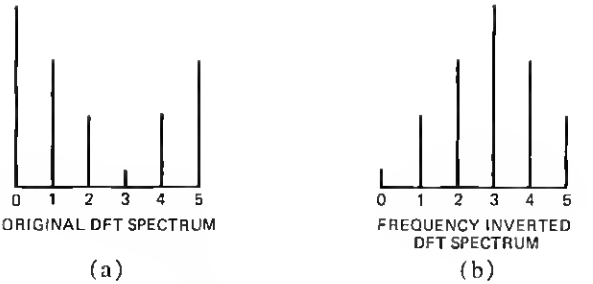


Fig. 8—Frequency inversion.

PAM speech samples. Although the analyses of preceding sections can be applied in principle to the special case of binary samples, the utility of such analyses may be limited because the issue of interest is the effect of scrambling on the decoded speech spectrum rather than the spectrum of the bits itself. Furthermore, there are many other factors, such as the varying significance of PCM (or DPCM) bits depending on their position in a PCM (or DPCM) word. In view of the above problems, we have deferred our observations on speech code scrambling to the section on perceptual experiments (Section IV), rather than attempt analytical predictions. We briefly analyze, on the other hand, the interesting case of permutations with sign changes, before proceeding to summarize results of perceptual experiments.

III. TRANSFORMATIONS WITH SIGN CHANGES—FREQUENCY INVERSION

The total number of permutations of a sequence of N samples is $N!$. If sign changes are allowed in addition to position changes, the total number of possible transformations increases to $2^N N!$. The analytical approaches of Section II can perhaps be extended to study such transformations. The purpose of this section, however, is only to consider a very specific transformation. This transformation involves no explicit permutation, but it introduces sign changes in every other input sample. This is equivalent to the classical encryption technique of analog frequency inversion.[1]

We begin by recapitulating that DFT inversion, as given by (11a) and (11b), does not constitute analog frequency inversion, because the DFT of a real signal is symmetric in the sense $|X(r)| = |X(N-r)|$ for $r = 1, 2, \cdots N$, and the highest analog frequency corresponds to $[N/2]$ where $[\ ]$ indicates "largest integer in." Analog frequency inversion would result, on the other hand, if the DFT coefficients are inverted about $N/4$ (Fig. 8), assuming that N is even. (Such inversion is achieved if all the DFT coefficients are translated cyclically through a distance of $N/2$. For

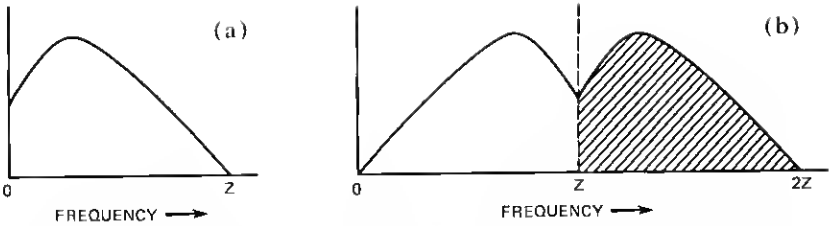


Fig. 9—Frequency inversion by modulation and low-pass filtering. (a) Original band-limited spectrum. (b) Result of modulating spectrum with carrier at Z. The left half of (b) is the inverse of (a).

odd N, there is a modification of these coefficients as well, since a simple translation will render the spectrum asymmetric.)

For a cyclic shift through $N/2$, the N by N frequency transformation matrix (for even N) is

$$T = \begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix}, \tag{18a}$$

where the submatrices are each of order $N/2$. From (8b), the corresponding input transformation is

$$P_{\text{FREQ INVERSION}} = \begin{bmatrix} 1 & 0 & 0 & 0 & . & . \\ 0 & -1 & 0 & 0 & & \\ 0 & 0 & 1 & 0 & & \\ 0 & 0 & 0 & -1 & & \\ . & & & & . & \\ . & & & & . & \end{bmatrix}. \tag{18b}$$

Thus, analog frequency inversion is obtained if alternate input samples are multiplied by -1. This is also apparent from the fact that frequency inversion occurs when a band-limited signal is modulated by a carrier at the highest input frequency, and out-of-band frequencies are filtered out from the modulation product (see Fig. 9). In the digital technique, the sequence of $+1$ and -1 elements in T corresponds to the carrier, and its frequency is simply the frequency of $+1$ (or -1) entries.

IV. EXPERIMENTAL RESULTS

In this section, we summarize experimental observations on speech waveform encryption using PAM samples as well as APCM, ADPCM, and ADM codes.[3] The letter A in code notation stands for instantaneously adaptive quantization,[3] and our DPCM and DM codes used simple first-order predictors. Our results are from computer simulations, and the studies have included PR and U permutations, analog frequency inversion, and (the academic case of) DFT inversion. Our conclusions are based on informal perceptual tests and on spectrograms of original and en-

crypted speech. The speech sample used in the experiments was a 2-second band-limited (200 to 3200 Hz) utterance, "The chairman cast three votes."

4.1 Perceptual observations

4.1.1 Scrambling with PR and U permutations

The sampling frequency for PAM and ADM bits was 24 kHz. The APCM and ADPCM bits used Nyquist-sampled (8 kHz) speech with 3 bits of quantization per sample. Thus, the scramblers operated on 24-kHz sequences in all four cases, and the encoding delay due to scrambling was the same for all the four waveform formats as long as the scrambler block length N was the same. Values of N ranged from 4 to 64. The speech-encoding qualities resulting in the identical bit-rate (24 kb/s) ADM, APCM, and ADPCM codes were, of course, noticeably different,[3] but this issue is not strictly relevant to the present discussion. The following were the main observations on encryption efficiency.

(*i*) In comparing pseudorandom and uniform permutations, we did not notice important perceptual differences: the U-permutation speech scramblers were about as good as the PR scramblers for given N, and, in fact, they were slightly better in some cases than PR scramblers.

(*ii*) In comparing speech code formats from the point of view of the benefits of scrambling, there was a definite ordering of efficiency:

$$\text{PAM} < \text{ADM} < \text{APCM} < \text{ADPCM}. \tag{19}$$

This means that for a given encoding delay (or scrambling block length N), the least desirable candidate for scrambling is a PAM sequence, while the most desirable format was an ADPCM code. The following specific observations are worth noting:

(a) Scrambling of PAM samples is marginally effective with $N = 64$.

(b) Scrambling of ADM samples is quite effective with $N = 64$. In fact, for very casual encryption using ADM bit-scrambling, even $N = 32$ can be useful; with $N = 32$, the specific U permutations that were used ($k_1 = 7$) were slightly more effective than the PR permutations.

(c) Scrambling of APCM and ADPCM bits destroys speech intelligibility very effectively with $N = 32$, while for casual encryption even $N = 16$ can be useful. Thus, ADPCM at 24 kb/s, with a scrambling delay of 16/24 = 0.67 ms, can be a very attractive candidate for practical speech-privacy systems (for example, optional facilities for privacy in mobile telephony).

4.1.2 Frequency inversion

The speech formats used here were the same as for the scrambler studies, except in the case of PAM samples. These were sampled at the

Nyquist frequency of 8 kHz (instead of 24 kHz) to provide frequency inversion (Section III and Fig. 9), which is a classical speech-encryption procedure.[1] As mentioned in Section III, the inversion of every other Nyquist-rate PAM sample provides a simple digital technique for speech frequency inversion. The sign-reversal operations on ADM, APCM, and ADPCM codes are less interesting in that they do not provide speech spectrum inversion, but only the inversion of respective bit spectra. In any case, sign reversals of adjacent samples provided effective speech distortions for all the speech-formats studies, with the following ordering of encryption efficiency in informal tests:

$$\text{ADM} < \text{PAM} < \text{APCM} < \text{ADPCM}. \tag{20}$$

The important point, however, is that unlike in scrambling, encryption potential in frequency inversion cannot be increased by such simple means as increasing a block length N ($N = 2$ is the minimum as well as maximum useful block length for frequency inversion). More importantly, owing to the fact that only one key is associated with this technique, frequency inversion is not suitable for formal encryption with time-varying keys. Furthermore, even for casual privacy, the aforementioned utility of frequency inversion (with PAM samples) is to be qualified with the observation that listeners can be trained to follow frequency-inverted speech. Frequency inversions with ADM bits were rather ineffective, for reasons not completely understood by the authors; while the more effective APCM and ADPCM inversions (20) were no more efficient than APCM- and ADPCM-based scramblers. (See, for example, spectrograms in Figs. 10a and b.)

Finally, perceptual tests confirmed our earlier observation that DFT inversions are academic and useless for speech encryption.

4.2 Speech spectrograms

Figure 10 reinforces and supplements the perceptual tests summarized in the previous section.

Figure 10a compares the classical technique of frequency inversion with a scheme that inverts the signs of alternate ADPCM bits. Notice the lack of speech-like patterns in the latter example and the contributions of ADPCM quantization noise that also serve to reduce speech intelligibility. In contrast, the frequency inversion of PAM samples leads to a spectrogram that is informationally equivalent to the original speech spectrogram, being only a mirror image of it across the 4-kHz line.

Figure 10b shows the benefits of increasing the block length N in temporal scrambling for the example of ADPCM codes and PR permutations.

Figure 10c demonstrates the effect of U permutations in scrambling, and compares PAM, ADM, and ADPCM samples as candidates for

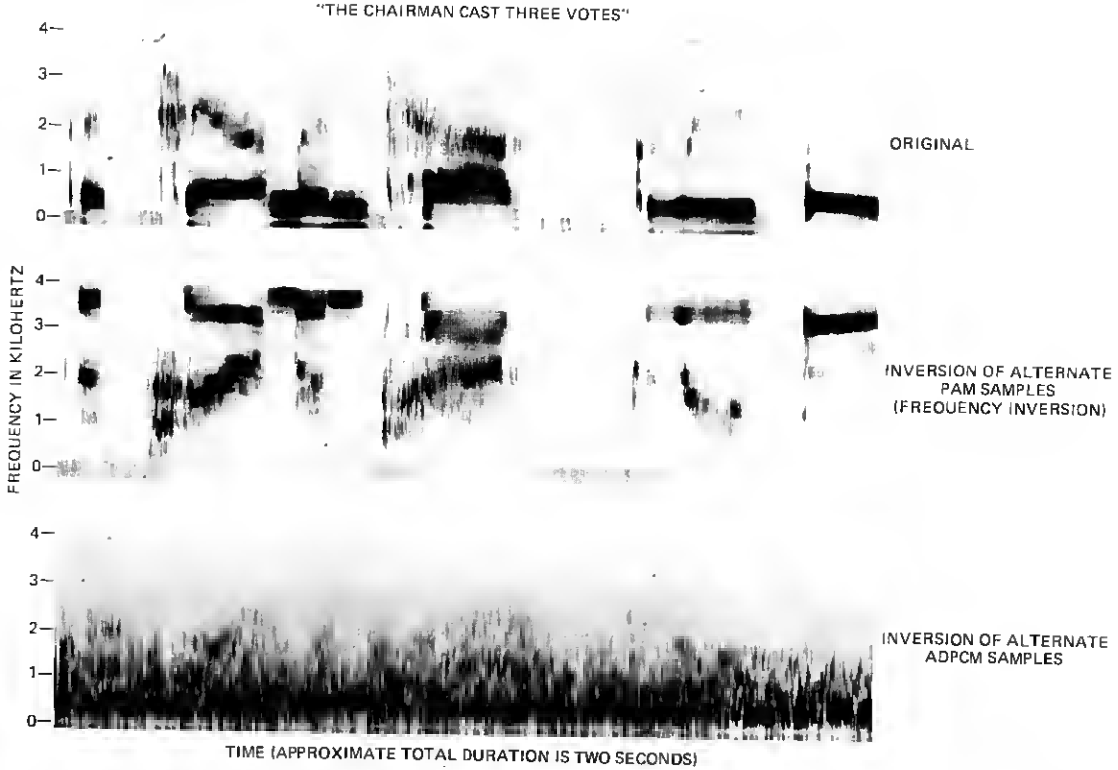


Fig. 10(a)—Spectograms of original and scrambled speech: original speech, frequency inversion, sign changing of alternate ADPCM bits.

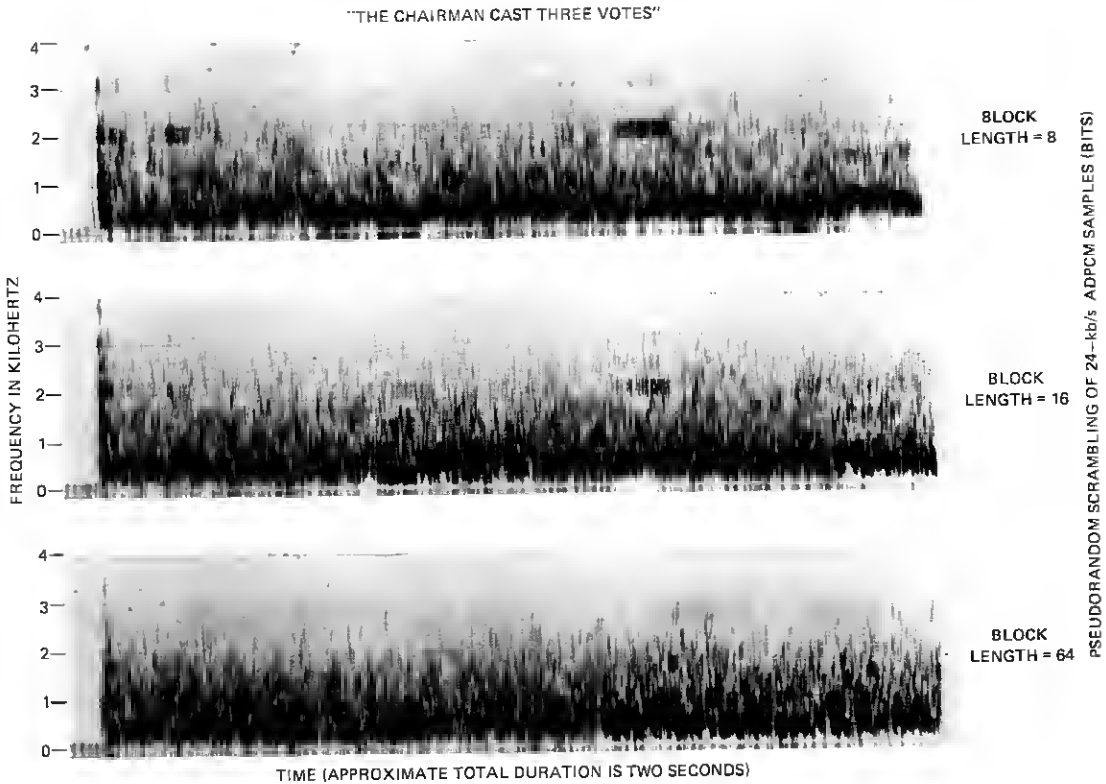


Fig. 10(b)—Spectograms of PR scrambling of ADPCM bits: N = 8, 16, 64.

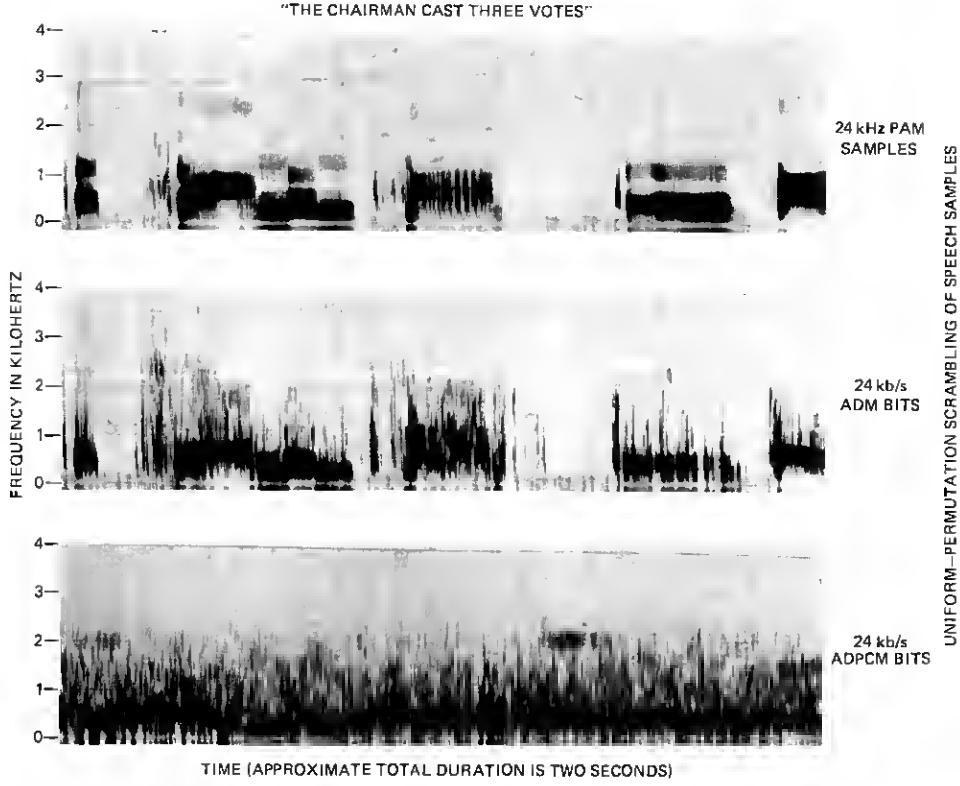


Fig. 10(c)—Spectograms of U scrambling of speech codes ($N = 16$): PAM, ADM, ADPCM.

scrambling. The spectrogram confirms the ordering of efficiency noted in (19): PAM < ADM < ADPCM.

V. ENCRYPTION, SCRAMBLING, AND MASKING

The preceding sections have looked mainly at effective privacy employing temporal permutations. The effectiveness of scrambling for encryption would depend on the rate at which the key is changed, the distortion produced by each key, and the number of keys available. Assuming that the distortion introduced by a typical key is adequate, a measure of goodness of any subfamily of permutations would be the number of members of that set. On this count, U permutations are better than PR permutations ($K_U > K_{PR}$ for a given N, Table II). We do not have a sufficient understanding of how many of these K_U or K_{PR} permutations are desirable from the point of view of distorting speech.* But we believe that for both PR and U permutations, the numbers of trivial or obviously useless permutations are small fractions of the number of possibilities in Table II. It can also be shown that the degree of complexity in encryption is equal to $\log_2 K$. This number results from statistical procedures that minimize the number of receivers needed for cryptanalysis.[4]

Many so-called speech scramblers encrypt speech not by temporal permutations (as in our definition of scrambling), but by the addition of appropriate masking signals. A typical example is the EXCLUSIVE OR mod-2 addition of a pseudorandom binary sequence, bit by bit, to an ADM or PCM bit stream.

If the masking signal is such that the spectrum of the sum signal is white, we obtain a perfect cipher in some sense. However, the entropy of the key must equal the entropy of the signal to be encrypted, and a perfect cipher is an idealized case in this sense. Realistically, we seek masking signals that change slowly with the message waveform, and typical resultant spectra are non-speech-like although not perfectly white.

The use of modulo-m additions (example modulo-2 adds) in masking ensures that the amplitude range of the encrypted signal is the same as that of the input signal; preservation of properties such as dynamic range serves as a precaution against techniques of cryptanalysis.

Temporal scrambling has the following limitations with respect to masking: (*i*) scrambling introduces encoding delays ($= N$) and (*ii*) it leaves long ($\gg N$) tell-tale silences unaffected in speech encryption. Masking, on the other hand, is characterized by a greater complexity of

* This depends not only on the properties of the permutations, but also on those of the samples being permuted. Highly correlated PAM samples, for example, demand more drastic permutations than less redundant ADM bits for a given N.

the key signal. If delays of a few milliseconds are not objectionable and if the exposure of the on-off (speech-silence) patterns is not a problem, the simplicity of scrambling should make it more attractive.

A final issue is the comparison of temporal scrambling and masking purely from the point of view of reducing the intelligibility of speech sounds. In at least one experiment,[8] performances have been found to be very comparable. This experiment used 24-kb/s ADPCM speech bits. For scrambling, a block length of $N = 16$ was employed, and for masking, a 16-bit PR number (binary sequence) was added (modulo 2), bit-by-bit, to each of 16 contiguous ADPCM bits in the speech code to be masked.

For practical implementations, it is conceivable that sophisticated schemes may run permutation and masking operations in tandem. Also, if the associated keys are time-varying, it may be practical to use certain serial configurations for the shift register, as described in Ref. 9, in place of a bank of conventional hard-wired registers.

APPENDIX

Proof of Theorem (10)

Recall from (8a) that $T = FPF^{-1} = [FP]F^{-1}$. The value of T_{rs} is the dot-product of the sth column of $[F^{-1}]$ and the rth row of $[FP]$.

The sth column in F^{-1} has a typical element W^{-Rs}/N where R is the row number [see (9)].

To evaluate the dot product that gives T_{rs}, we need V_{rR}, the Rth column element in the rth row of $[FP]$. The quantity V_{rR} in turn is the product of the rth row in F and the Rth column in P. The only nonzero elements in P are the 1s that occur in positions $a, b, c, d, \cdots$ in rows 0, 1, 2, 3, $\cdots$ as characterized by $P[0 \rightarrow a, 1 \rightarrow b, 2 \rightarrow c, 3 \rightarrow d, \cdots]$. Furthermore, the rth row in F has a typical element W^{rn}, where n is the column number [see (5)]. Consequently, V_{rR} has the form $W^{rn(R)}$, where $n(R)$ is the column element in the rth row of F, which gets multiplied by the single 1 entry in the Rth column of P. (The other elements in the rth row of F get multiplied by 0 entries in the Rth column of P, and do not contribute to V_{rR}.)

Consequently, the dot product that specifies T_{rs} has the form

$$\frac{1}{N}\sum_{R=0}^{N-1} W^{-Rs}V_{rR} = \frac{1}{N}\sum_{R=0}^{N-1} W^{-Rs}W^{rn(R)}.$$

Because of a one-to-one mapping between R and $n(R)$, the above summation can be rewritten as a summation over $n(R)$:

$$T_{rs} = \frac{1}{N}\sum_{n(R)=0}^{N-1} W^{n(R)\cdot(r)-(R\cdot s)}.$$

In characterizing P, $n(R)$ values of 0, 1, 2, 3, $\cdots$ correspond to R values

of $a, b, c, d, \cdots$. Hence,

$$T_{rs} = \frac{1}{N}(W^{-as} + W^{r-bs} + W^{2r-cs} + W^{3r-ds} + \cdots).$$

VI. ACKNOWLEDGMENTS

The authors would like to thank L. R. Rabiner and J. L. Flanagan for useful consultations.

REFERENCES

1. D. Kahn, *The Code-Breakers*, New York: The Macmillan Company, 1967.
2. M. R. Sambur and N. S. Jayant, "Speech Encryption by Manipulations of LPC Parameters," B.S.T.J., *55*, No. 9 (November 1976), pp. 1373–1388.
3. N. S. Jayant, "Digital Coding of Speech Waveforms—PCM, DPCM and DM Quantizers," Proc. IEEE, *62* (May 1974), pp. 611–632.
4. C. E. Shannon, "Communication Theory of Secrecy Systems," B.S.T.J., *28*, No. 4 (October 1949), pp. 656–715.
5. S. W. Golomb, *Shift Register Sequences*, San Francisco: Holden-Day, 1967, pp. 62–65.
6. R. G. Gallager, *Information Theory and Reliable Communications*, New York: John Wiley, 1968.
7. F. J. MacWilliams and N. J. A. Sloane, "Pseudo-Random Sequences and Arrays," Proc. IEEE, *64* (December 1976), pp. 1715–1728.
8. N. S. Jayant, unpublished work.
9. S. V. Ahamed, "The Design and Embodiment of Magnetic-Domain Encoders and Single-Error-Correcting Decoders for Cyclic Block Codes," B.S.T.J., *51*, No. 2 (February 1972), pp. 461–485.